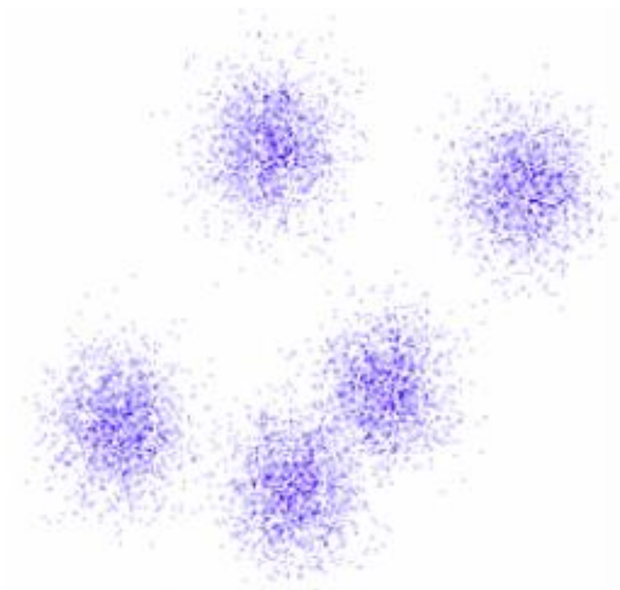
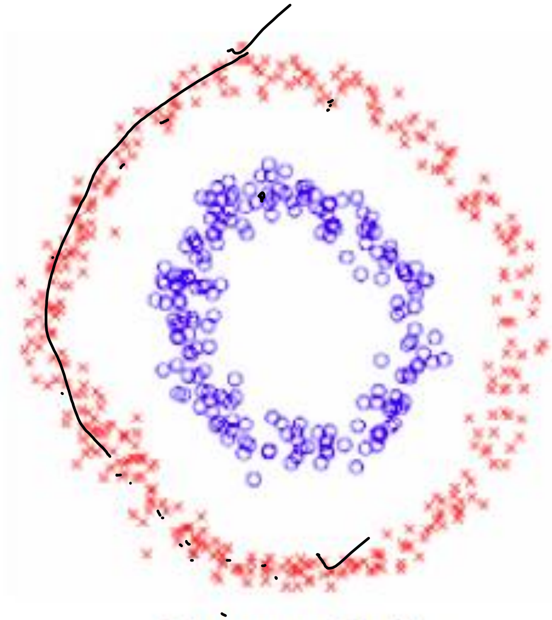
Data Clustering

- Two different criteria
  - Compactness, e.g., k-means, mixture models
  - Connectivity, e.g., spectral clustering



Compactness



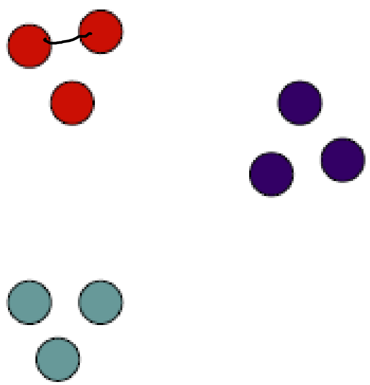
Connectivity

Graph Clustering

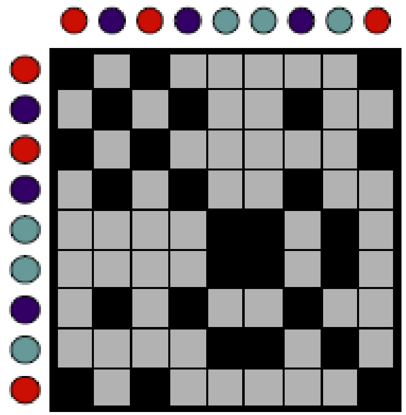
Goal: Given data points $X_1, \ldots, X_n$ and similarities $w(X_i,X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

Similarity Graph: G(V,E,W)

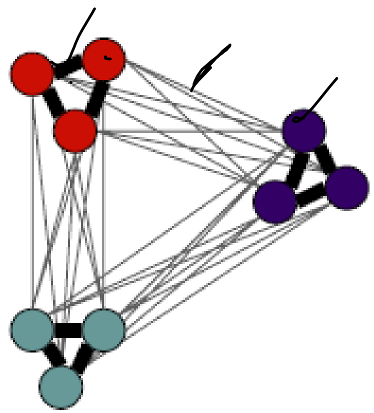
V – Vertices (Data points)
E – Edge if similarity > 0
W - Edge weights (similarities)



Data



Similarities



Similarity graph

Partition the graph so that edges within a group have large weights and edges across groups have small weights.

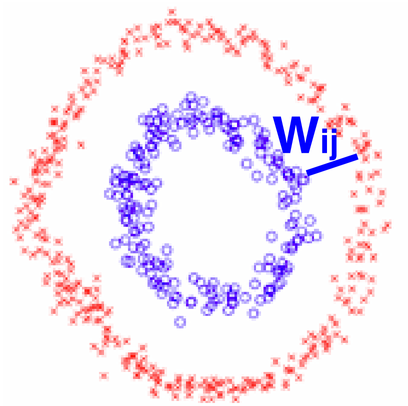
Similarity graph construction

Similarity Graphs: Model local neighborhood relations between data points

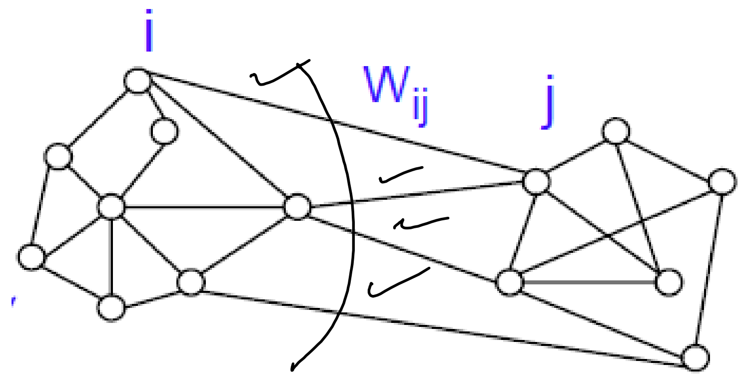
E.g. Gaussian kernel similarity function

$$\boxed{W_{ij} = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}}$$

Controls size of neighborhood



Data clustering

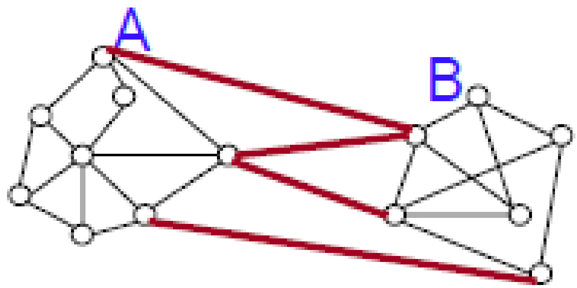


$G = \{V, E\}$

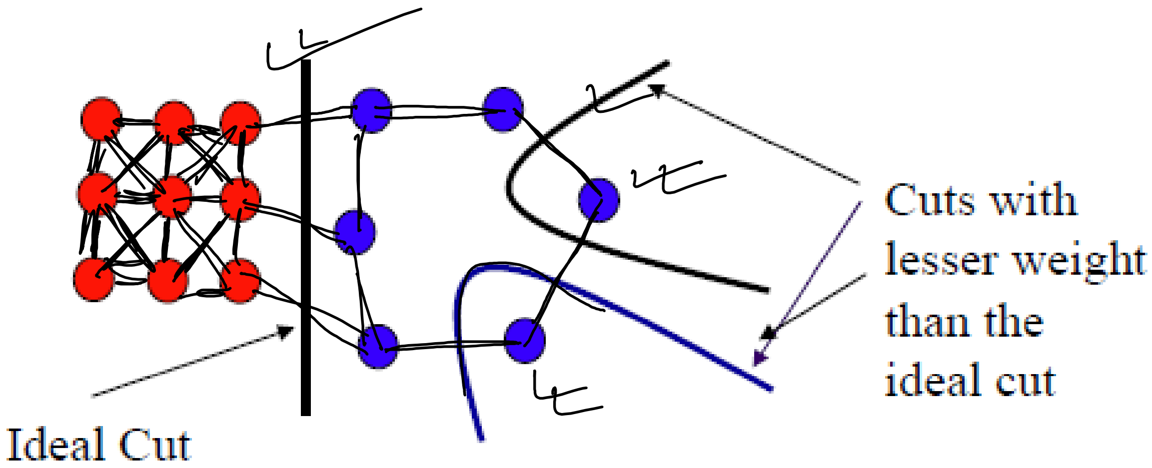
Partitioning a graph into two clusters

Min-cut: Partition graph into two sets A and B such that weight of edges connecting vertices in A to vertices in B is minimum.

$$\text{cut}(A, B) := \sum_{i \in A, j \in B} w_{ij}$$



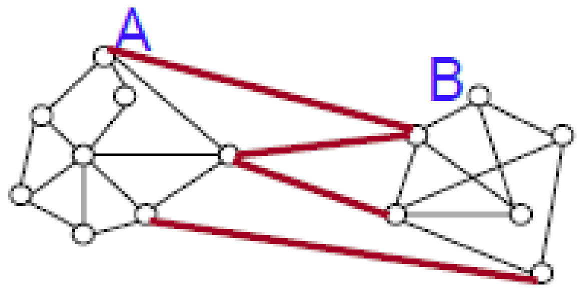
- Easy to solve O(VE) algorithm
- Not satisfactory partition – often isolates vertices



Partitioning a graph into two clusters

Partition graph into two sets A and B such that weight of edges connecting vertices in A to vertices in B is minimum & size of A and B are very similar.

$$\mathrm{cut}(A, B) := \sum_{i \in A, j \in B} w_{ij}$$



Normalized cut:

Find A and B with min

$$\mathrm{Ncut}(A, B) := \mathrm{cut}(A, B)\left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)}\right)$$

$$\mathrm{vol}(A) = \sum_{i \in A} d_i$$

But NP-hard to solve!!

Spectral clustering is a relaxation of these.

Finding Min cut.

labels = $\{1, -1\}$.

$f_i = 1$ if $x_i \in A$.

$f_i = -1$ if $x_i \in B$

$$\min_{f = \{f_1, f_2, \cdots f_n\}} \sum_{\substack{i \in A \\ j \in B}} w_{ij} = \frac{1}{4} \sum_{f_i \neq f_j} 4\, w_{ij}$$

$(f_i - f_j)^2 = 0$ if $f_i = f_j$

$(f_i - f_j)^2 = 4$ if $f_i \neq f_j$

$$= \frac{1}{4} \sum_{f_i = f_j} 0 \cdot w_{ij} + \frac{1}{4} \sum_{f_i \neq f_j} 4 \cdot w_{ij}.$$

$$= \boxed{\frac{1}{4} \sum_{i,j} (f_i - f_j)^2 w_{ij}}$$

$$\frac{1}{4}\sum_{ij}(f_i-f_j)^2 w_{ij} = \frac{1}{4}\sum_{ij} f_i^2 w_{ij} + f_j^2 w_{ij} - 2 f_i f_j w_{ij}$$

$$= \frac{1}{2}\left(\sum_{ij} f_i^2 w_{ij} - \sum_{ij} f_i f_j w_{ij}\right)$$

$$d_i = \sum_j w_{ij}$$

$$= \frac{1}{2}\sum_i f_i^2 d_i - \frac{1}{2}\sum_{ij} f_i f_j w_{ij}$$

$$D = \mathrm{dig}(d_i)$$

$$W = [w_{ij}]$$

$$= \frac{1}{2} f^T D f - \frac{1}{2} f^T W f \qquad f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \end{bmatrix}$$

$$= \frac{1}{2} f^T \underline{(D - W)} f$$

$$= \frac{1}{2} f^T L f$$

$$\boxed{\min_f f^T \underline{L} f}$$

$$\boxed{\begin{matrix} D^{-\frac{1}{2}} L D^{-\frac{1}{2}} \\ D^{-1} L \end{matrix}}$$

Normalized Cut and Graph Laplacian

$$\text{Ncut}(A,B) := \text{cut}(A,B)\left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)}\right)$$

Let $\mathbf{f} = [f_1\ f_2\ \dots\ f_n]^T$ with $f_i = \begin{cases} \frac{1}{\text{vol}(A)} & \text{if } i \in A \\ -\frac{1}{\text{vol}(B)} & \text{if } i \in B \end{cases}$

$$\boxed{\mathbf{f}^T\mathbf{L}\mathbf{f} = \sum_{ij} w_{ij}(\mathbf{f}_i - \mathbf{f}_j)^2} = \sum_{i\in A, j\in B} w_{ij}\left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)}\right)^2$$

$$\mathbf{f}^T\mathbf{D}\mathbf{f} = \sum_j d_i\mathbf{f}_i^2 = \sum_{i\in A}\frac{d_i}{\text{vol}(A)^2} + \sum_{j\in B}\frac{d_i}{\text{vol}(B)^2} = \boxed{\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)}}$$

$$\boxed{\text{Ncut}(A,B) = \frac{\mathbf{f}^T\mathbf{L}\mathbf{f}}{\mathbf{f}^T\mathbf{D}\mathbf{f}}} = f^T L_{nor} f$$

$L_{nor} = D^{-1} L$

$D^{-1/2} L D^{-1/2}$

Normalized Cut and Graph Laplacian

$$\min \; \mathrm{Ncut}(A, B) = \min \frac{\mathbf{f}^T \mathbf{L} \mathbf{f}}{\mathbf{f}^T \mathbf{D} \mathbf{f}}$$

where $f = [f_1 \; f_2 \; \dots \; f_n]^T$ with $f_i = \begin{cases} \dfrac{1}{\mathrm{vol}(A)} & \text{if } i \in A \\ -\dfrac{1}{\mathrm{vol}(B)} & \text{if } i \in B \end{cases}$

Relaxation: $\boxed{\min \; \dfrac{\mathbf{f}^T \mathbf{L} \mathbf{f}}{\mathbf{f}^T \mathbf{D} \mathbf{f}}}$ s.t. $\boxed{f^T D 1 = 0}$

Solution: f – second eigenvector of generalized eval problem

$$\boxed{\mathbf{L}\mathbf{f} = \lambda \mathbf{D} \mathbf{f}}$$

Obtain cluster assignments by thresholding f at 0

Approximation of Normalized cut

$$\mathrm{Ncut}(A,B) := \mathrm{cut}(A,B)\left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)}\right)$$

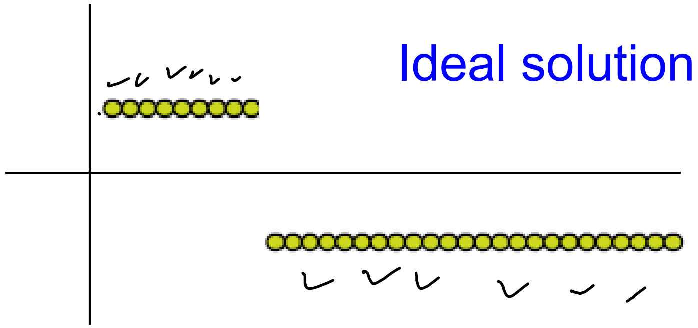
Let *f* be the eigenvector corresponding to the second smallest eval of the generalized eval problem.

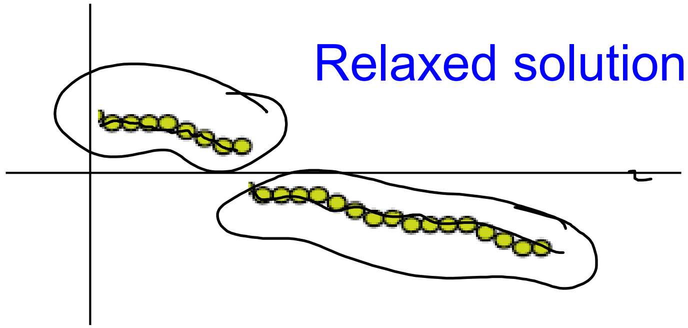
$$\boxed{\mathrm{Lf} = \lambda \mathrm{Df}}$$

Equivalent to eigenvector corresponding to the second smallest eval of the normalized Laplacian $L' = D^{-1}L = I - D^{-1}W$

Recover binary partition as follows:

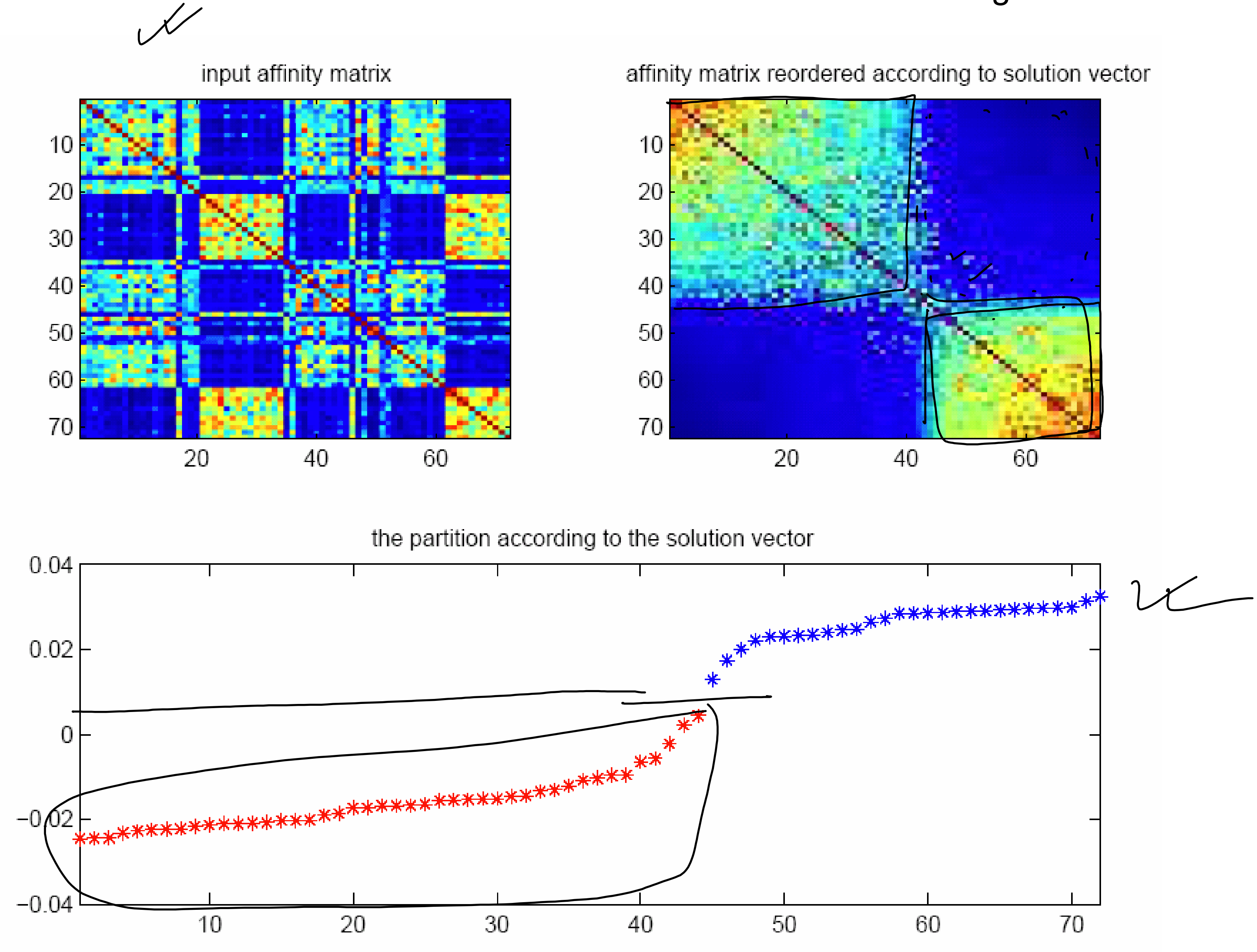
$i \in A$	*if*	$f_i \geq 0$
$i \in B$	*if*	$f_i < 0$





Example

Xing et al 2001



How to partition a graph into k clusters?

Spectral Clustering Algorithm

Input: Similarity matrix W, number k of clusters to construct

- Build similarity graph
- Compute the first k eigenvectors $v_1, \ldots, v_k$ of the matrix

$$\begin{cases} L & \text{for unnormalized spectral clustering} \\ L' & \text{for normalized spectral clustering} \end{cases}$$

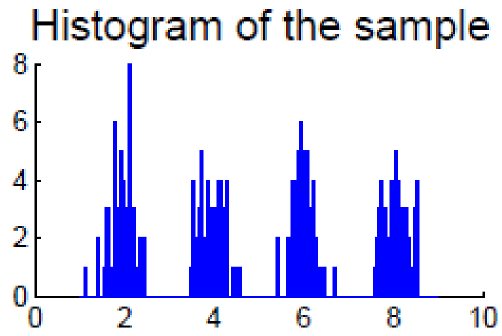
- Build the matrix $V \in \mathbb{R}^{n \times k}$ with the eigenvectors as columns
- Interpret the rows of V as new data points $Z_i \in \mathbb{R}^k$

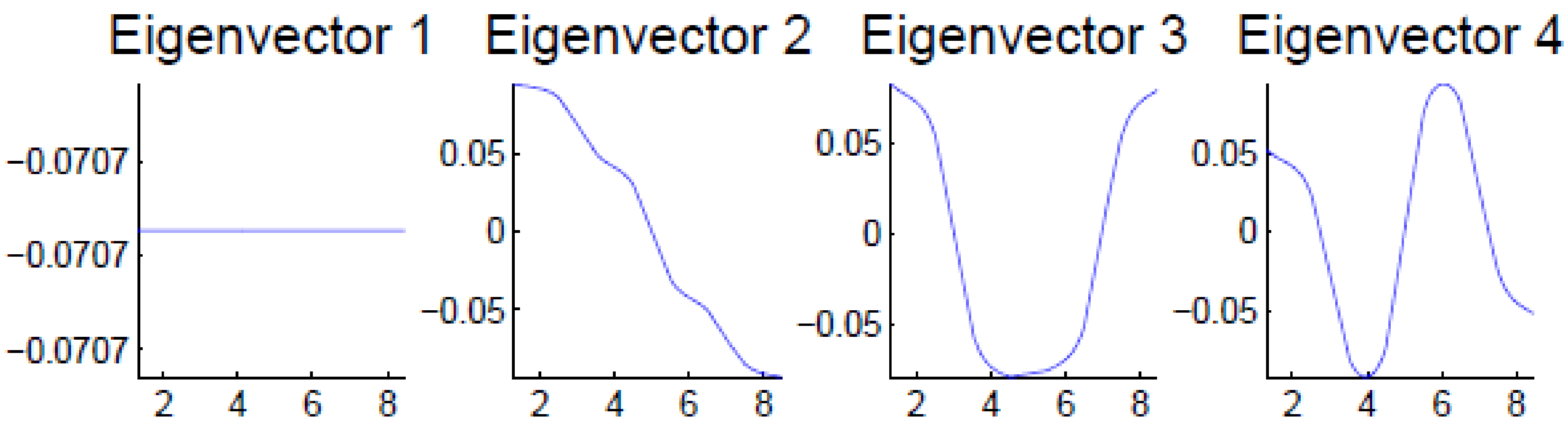
	v_1	v_2	v_3
Z_1	v_{11}	v_{12}	v_{13}
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Z_n	v_{n1}	v_{n2}	v_{n3}

Dimensionality Reduction
n x n $\rightarrow$ n x k

- Cluster the points Z_i with the k-means algorithm in $\mathbb{R}^k$.

Eigenvectors of Graph Laplacian



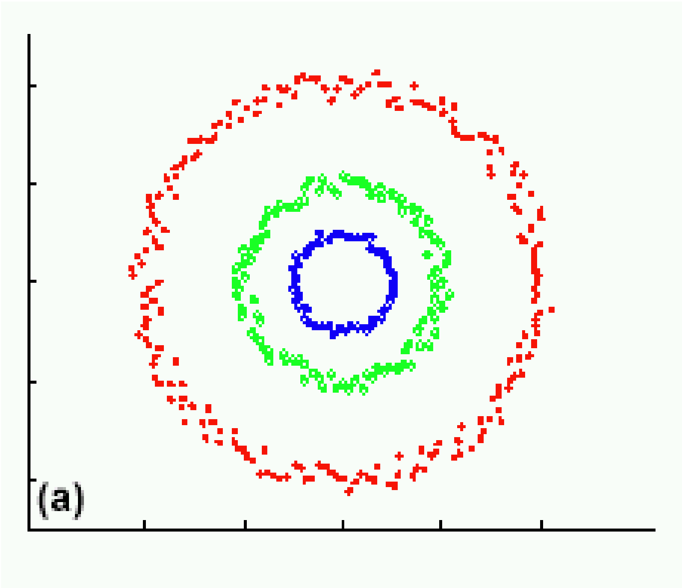


- 1st Eigenvector is the all ones vector ***1*** (if graph is connected)
- 2nd Eigenvector thresholded at 0 separates first two clusters from last two
- k-means clustering of the 4 eigenvectors identifies all clusters

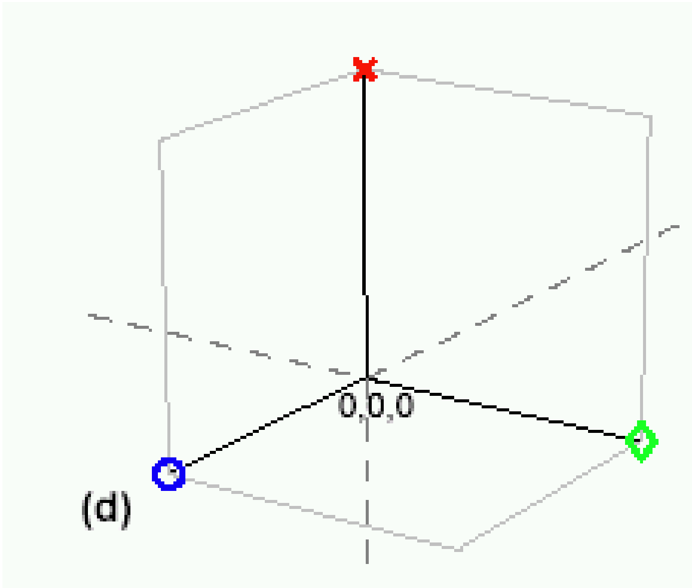
Why does it work?

Data are projected into a lower-dimensional space (the spectral/eigenvector domain) where they are easily separable, say using k-means.

Original data



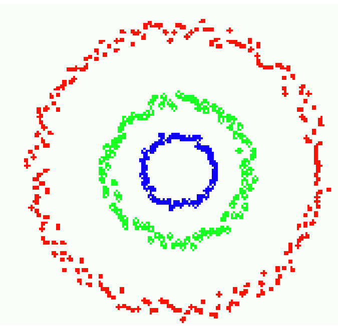
Projected data



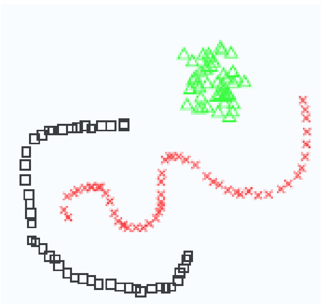
Graph has 3 connected components – first three eigenvectors are constant (all ones) on each component.

Understanding Spectral Clustering

- If graph is connected, first Laplacian evec is constant (all 1s)
- If graph is disconnected (k connected components), Laplacian is block diagonal and first k Laplacian evecs are:



OR



$$L = \begin{pmatrix} L_1 & & 0 \\ & L_2 & \\ 0 & & L_3 \end{pmatrix}$$

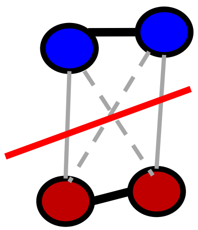
$$\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

First three eigenvectors

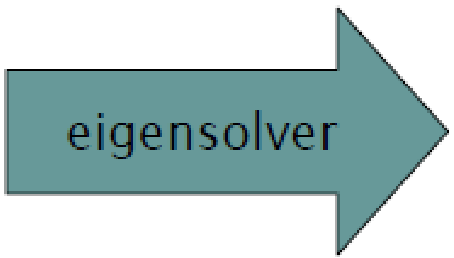
Understanding Spectral Clustering

- Is all hope lost if clusters don't correspond to connected components of graph? No!
- If clusters are connected loosely (small off-block diagonal enteries), then 1st Laplacian even is all 1s, but second evec gets first cut (min normalized cut)

$$\mathrm{Ncut}(A, B) := \mathrm{cut}(A, B)\left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)}\right)$$



1	1	.2	0
1	1	0	.1
.2	0	1	1
0	.1	1	1



.50
.50
.50
.50

.47
.52
-.47
-.52

1st evec is constant since graph is connected

Sign of 2nd evec indicates blocks

Why does it work?

Block weight matrix (disconnected graph) results in block eigenvectors:

$$\underbrace{\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}}_{W} \xrightarrow{\text{eigensolver}} \underbrace{\begin{bmatrix} .71 \\ .71 \\ 0 \\ 0 \end{bmatrix}}_{f_1} \quad \underbrace{\begin{bmatrix} 0 \\ 0 \\ .71 \\ .71 \end{bmatrix}}_{f_2}$$

Normalized to have unit norm

Slight perturbation does not change span of eigenvectors significantly:

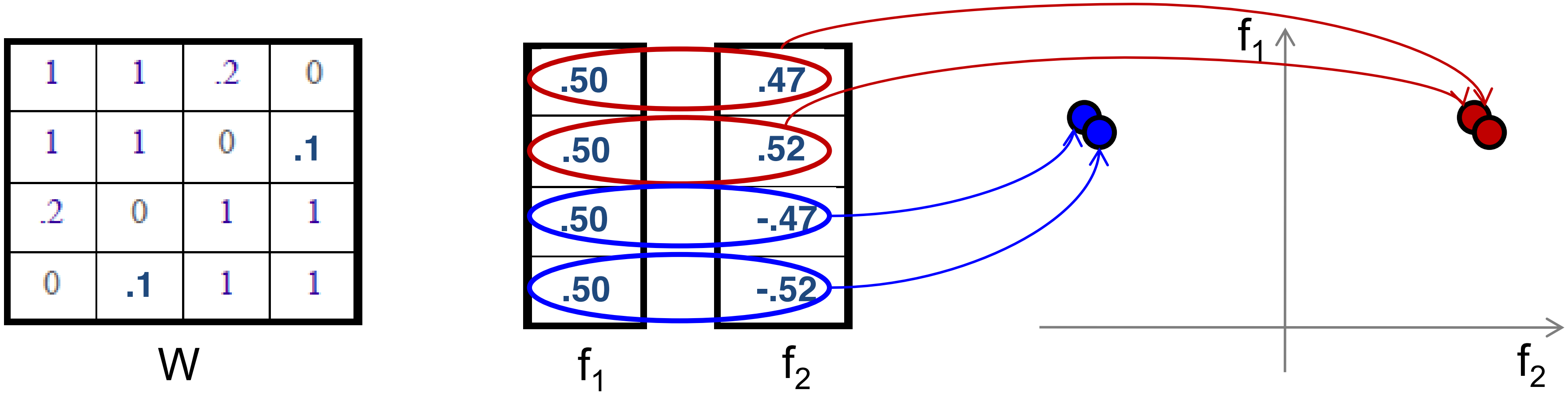
$$\begin{bmatrix} 1 & 1 & .2 & 0 \\ 1 & 1 & 0 & .1 \\ .2 & 0 & 1 & 1 \\ 0 & .1 & 1 & 1 \end{bmatrix} \xrightarrow{\text{eigensolver}} \begin{bmatrix} .50 \\ .50 \\ .50 \\ .50 \end{bmatrix} \quad \begin{bmatrix} .47 \\ .52 \\ -.47 \\ -.52 \end{bmatrix}$$

1st evec is constant since graph is connected

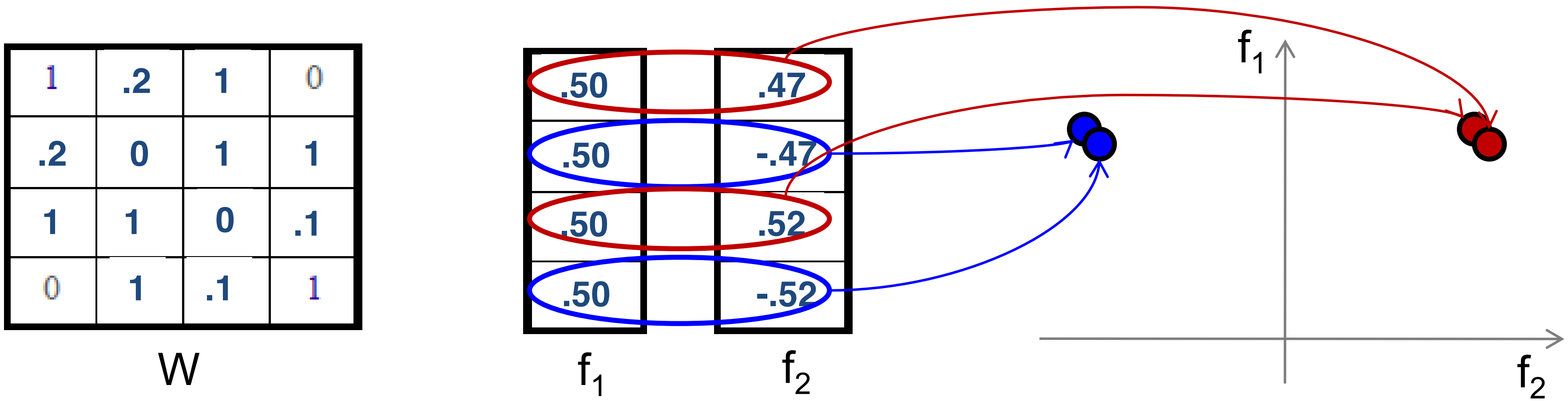
Sign of 2nd evec indicates blocks

Why does it work?

Can put data points into blocks using eigenvectors:



Embedding is same regardless of data ordering:



Understanding Spectral Clustering

- Is all hope lost if clusters don't correspond to connected components of graph? No!
- If clusters are connected loosely (small off-block diagonal enteries), then 1st Laplacian even is all 1s, but second evec gets first cut (min normalized cut)

$$\mathrm{Ncut}(A, B) := \mathrm{cut}(A, B)\left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)}\right)$$

- What about more than two clusters?

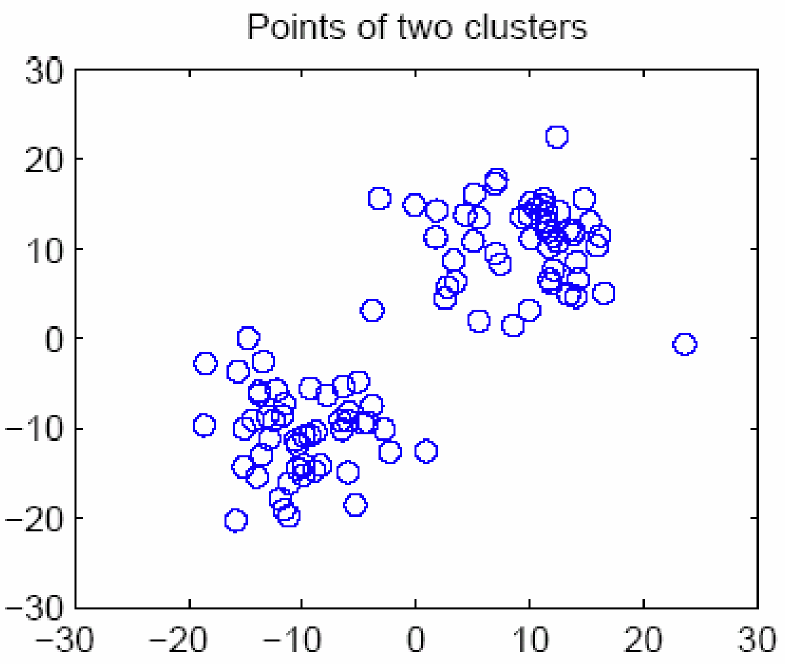
  eigenvectors $f_2, \ldots, f_{k+1}$ are solutions of following normalized cut:

$$\mathrm{Ncut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{\mathrm{cut}(A_i, \overline{A}_i)}{\mathrm{vol}(A_i)}$$

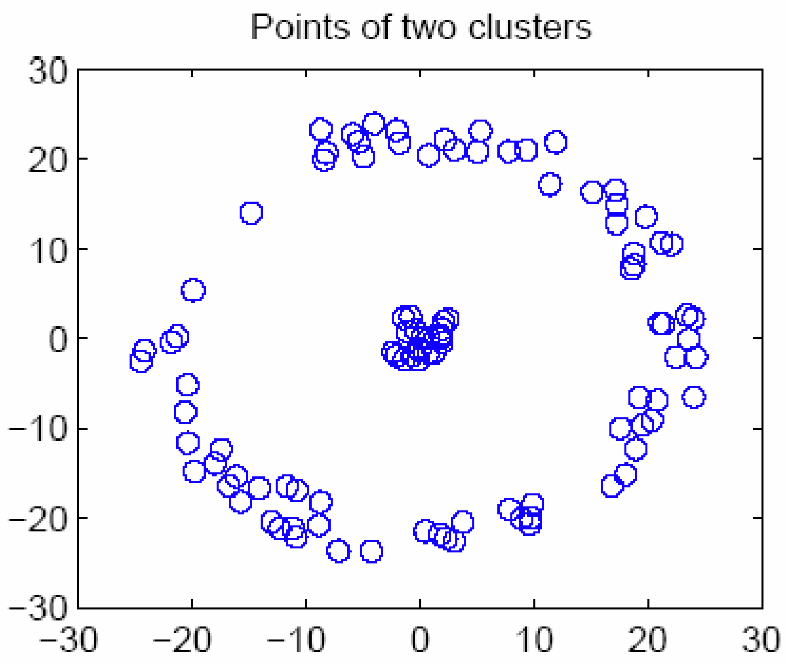
Demo: http://www.ml.uni-saarland.de/GraphDemo/DemoSpectralClustering.html

k-means vs Spectral clustering

Applying k-means to laplacian eigenvectors allows us to find cluster with non-convex boundaries.



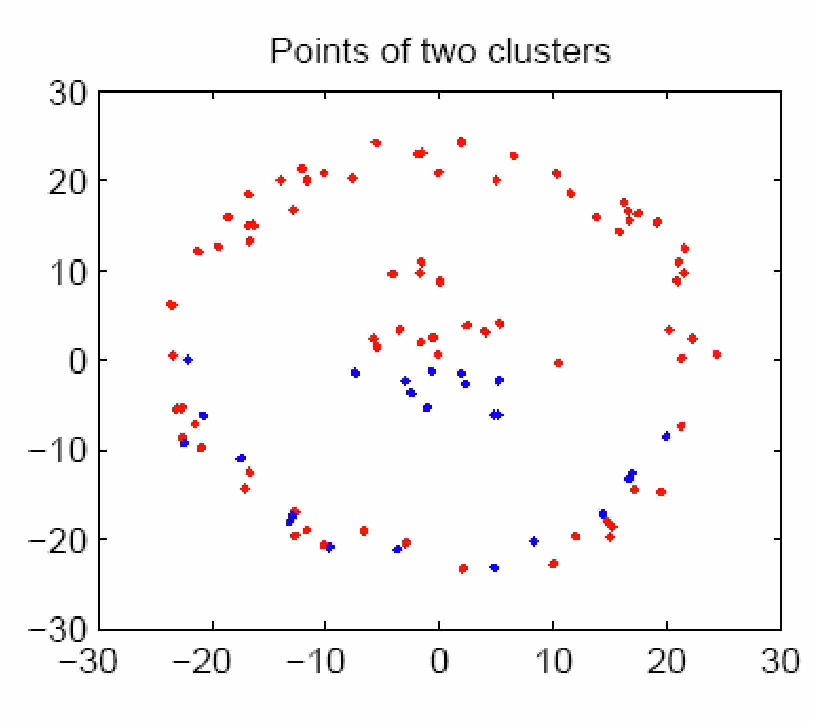
Both perform same



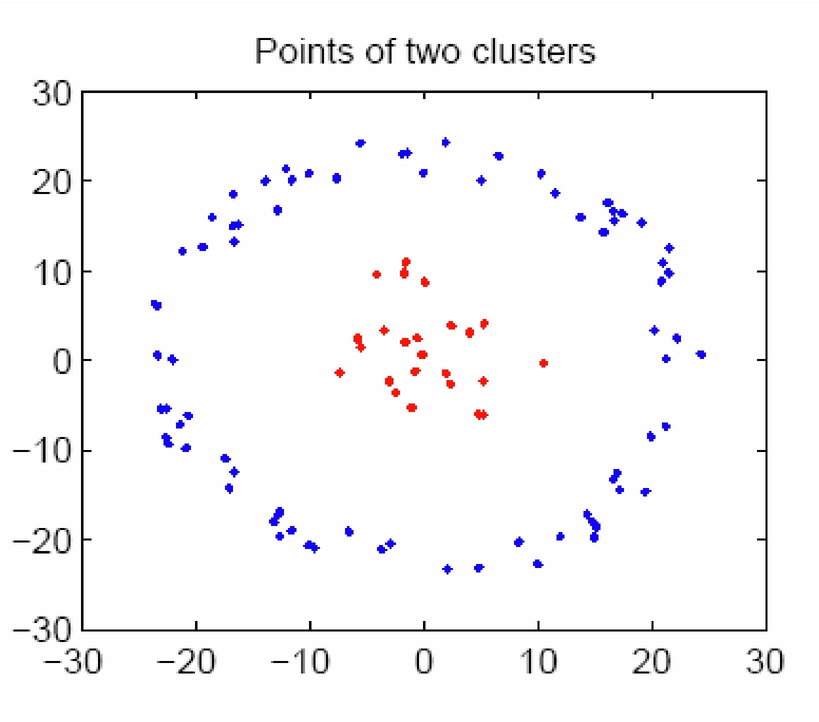
Spectral clustering is superior

k-means vs Spectral clustering

Applying k-means to laplacian eigenvectors allows us to find cluster with non-convex boundaries.



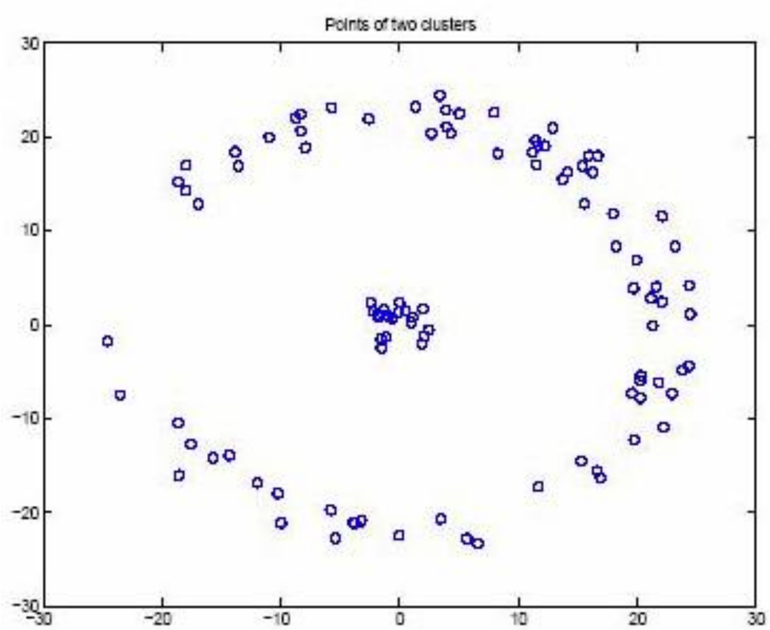
k-means output



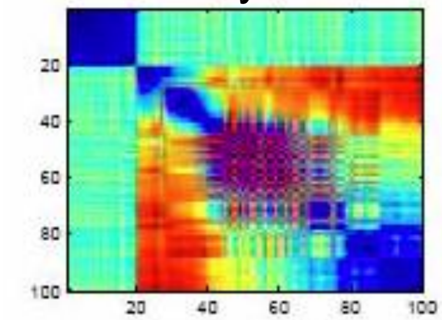
Spectral clustering output

k-means vs Spectral clustering

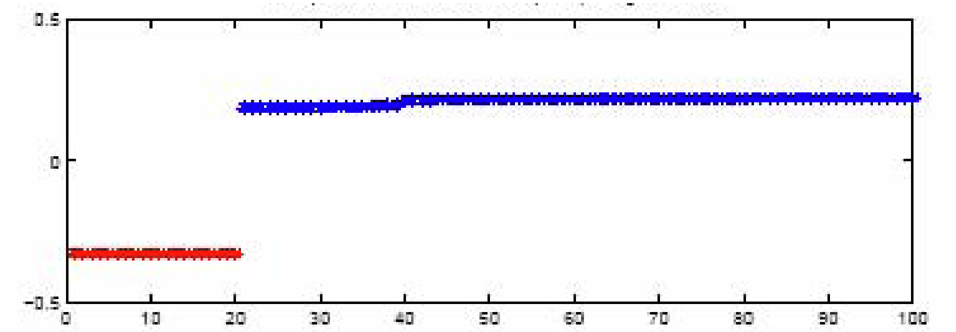
Applying k-means to laplacian eigenvectors allows us to find cluster with non-convex boundaries.



Similarity matrix

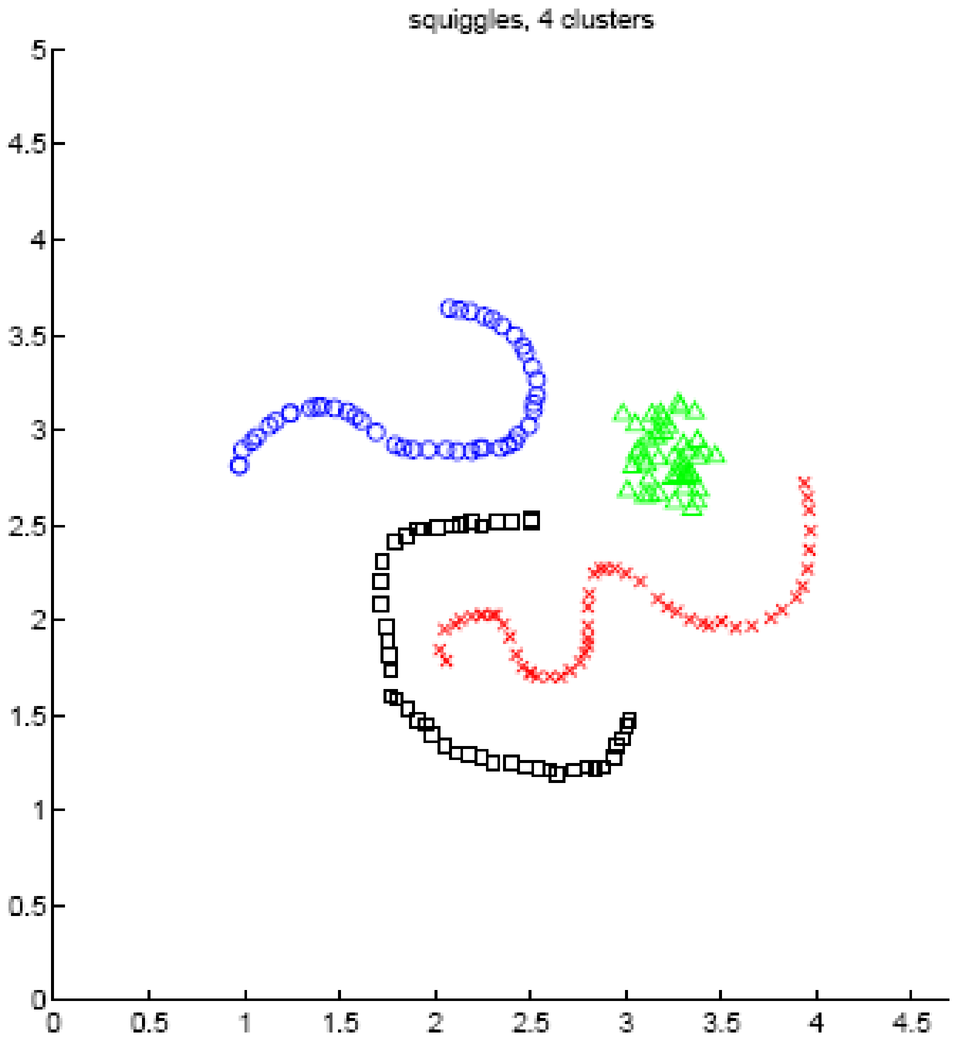


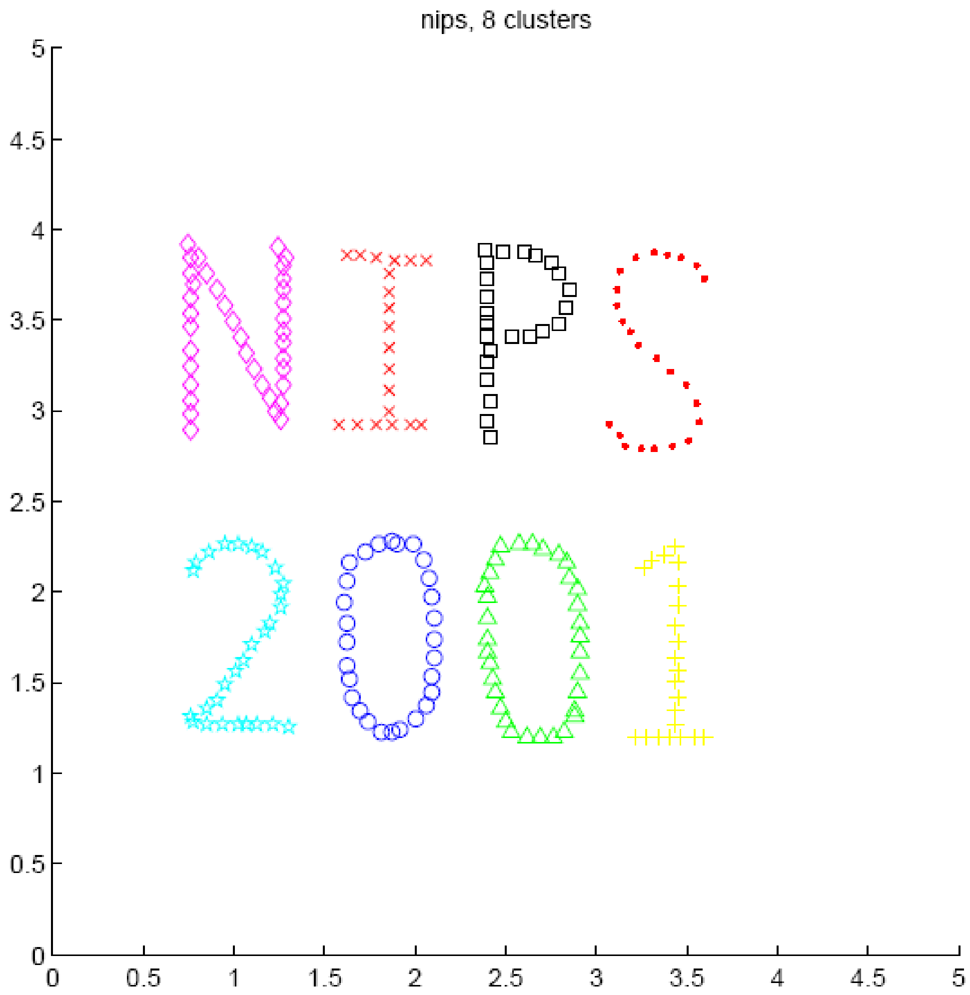
Second eigenvector of graph Laplacian



Examples

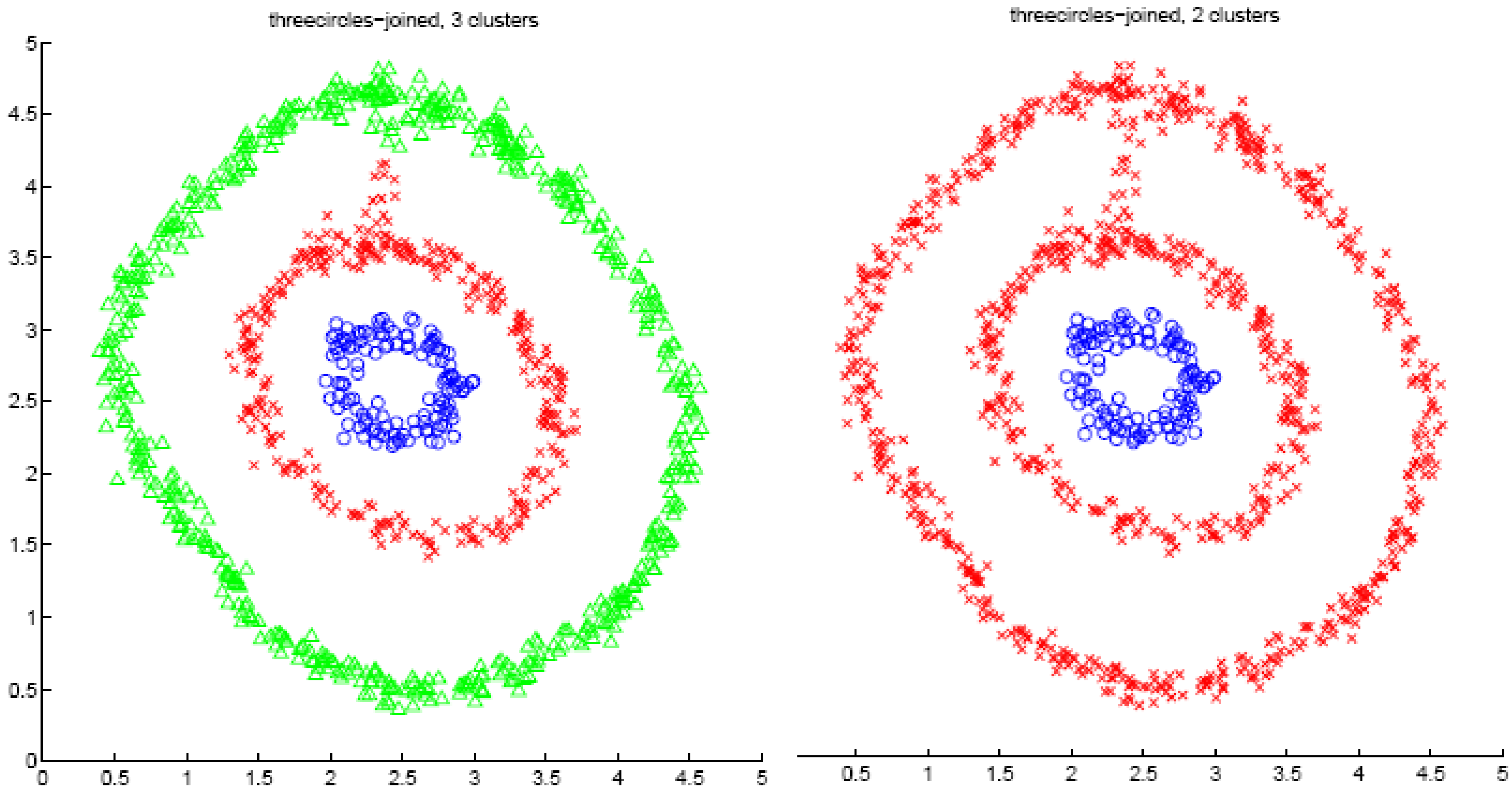
Ng et al 2001





Examples (Choice of k)

Ng et al 2001

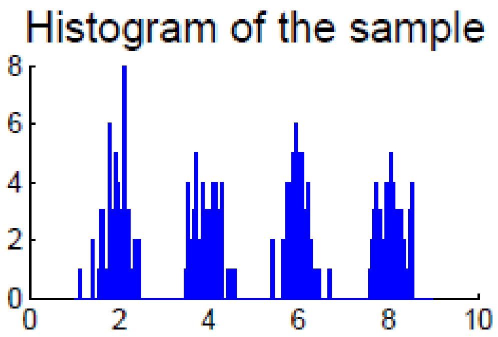


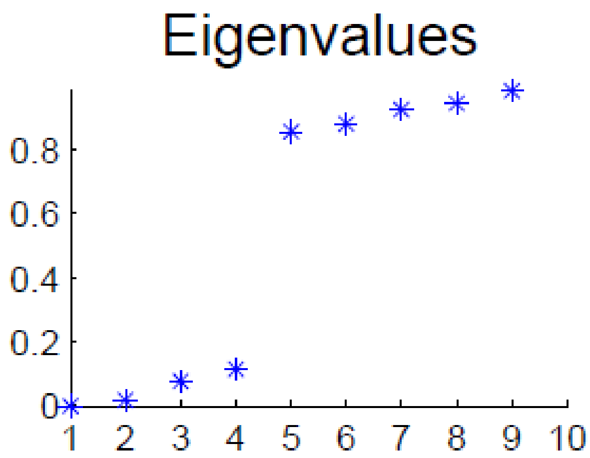
Some Issues

- Choice of number of clusters k

  Most stable clustering is usually given by the value of k that maximizes the eigengap (difference between consecutive eigenvalues)

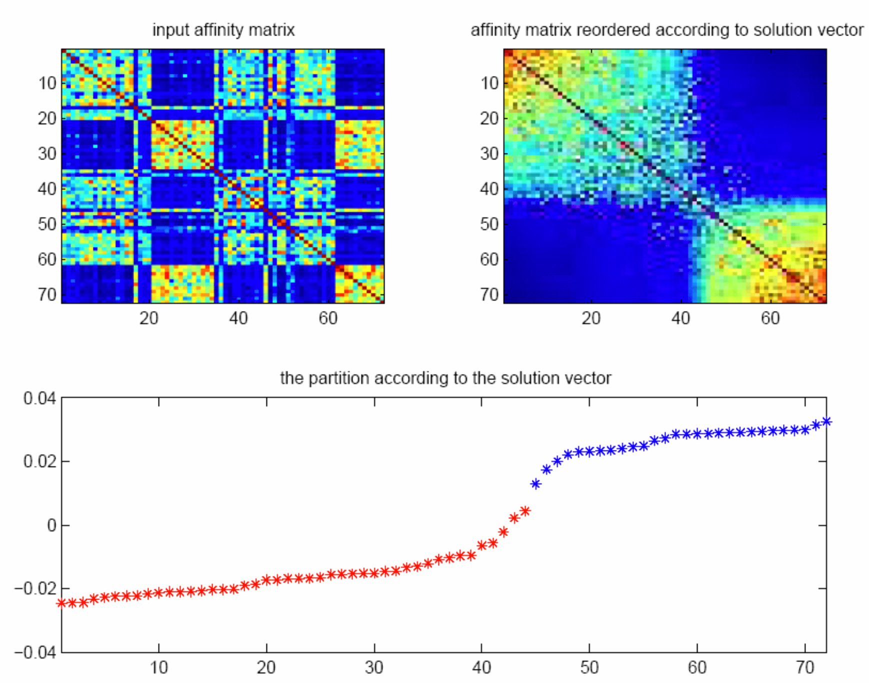
$$\Delta_k = \left| \lambda_k - \lambda_{k-1} \right|$$



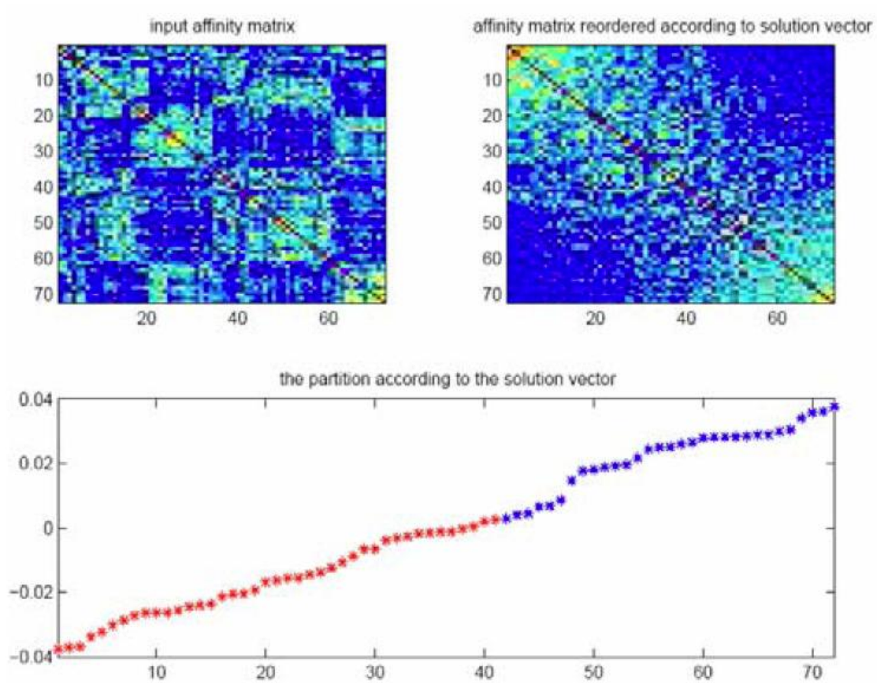


Some Issues

- Choice of number of clusters k
- Choice of similarity
  - choice of kernel
  - for Gaussian kernels, choice of σ



Good similarity measure



Poor similarity measure

Some Issues

- Choice of number of clusters k
- Choice of similarity
  - choice of kernel
  - for Gaussian kernels, choice of σ
- Choice of clustering method – k-way vs. recursive bipartite